VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

# Bakalářská práce

2016

Ondřej Gašpar

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Node editor v HTML5

# HTML5 Node Editor

2016

Ondřej Gašpar

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání bakalářské práce

| | |
|---|---|
| Student: | **Ondřej Gašpar** |
| Studijní program: | B2647 Informační a komunikační technologie |
| Studijní obor: | 2612R025 Informatika a výpočetní technika |
| Téma: | Node editor v HTML5<br>HTML5 Node Editor |
| Jazyk vypracování: | čeština |

Zásady pro vypracování:

Cílem práce je navrhnout a naimplementovat webovou aplikaci/komponentu, která bude plnit úlohu tzv. Node editoru s aktuálními možnostmi HTML5 a CSS3.

Body zadání:
1. Přehled existujících řešení, jejich vlastnosti a omezení ve vazbě na moderní trendy v HTML5.
2. Analýza a návrh vlastního řešení.
3. Implementace aplikace/komponenty s požadovanou funkcionalitou (editace, vizualizace, ukládání schémat, apod.).
4. Vytvoření experimentů a výkonnostních testů. Zohlednění optimalizace, jako např. shlukování.
5. Zhodnocení experimentů a dosažených výsledků.

Seznam doporučené odborné literatury:

[1] M. Pilgrim: HTML5: Up and Running, O'Reilly Media; 1 edition (August 24, 2010), ISBN-13: 978-0596806026
[2] D. Crockford: JavaScript: The Good Parts, O'Reilly Media; 1st edition (May 2008), ISBN-13: 978-0596517748

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Gajdoš, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

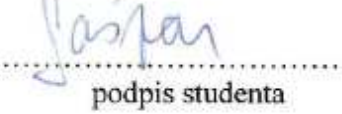

doc. Dr. Ing. Eduard Sojka
*vedoucí katedry*

prof. RNDr. Václav Snášel, CSc.
*děkan fakulty*

Prohlašuji, že

- jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu,
- byl jsem seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména §35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a §60 – školní dílo,
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně ke své vnitřní potřebě diplomovou práci užít (§35 odst. 3),
- souhlasím s tím, že jeden výtisk diplomové práce bude uložen v Ústřední knihovně VŠB-TUO k prezenčnímu nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že údaje o diplomové práci, obsažené v Záznamu o závěrečné práci, umístěném v příloze mé diplomové práce, budou zveřejněny v informačním systému VŠB-TUO,
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu §12 odst. 4 autorského zákona,
- bylo sjednáno, že užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše),
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě 26. dubna 2016



.....

podpis studenta

**Abstrakt**

Tato balakářská práce se zabývá návrhem a implementací webové aplikace / komponenty, která plní úlohu tzv. node editoru. Aplikace využívá aktuálních možností HTML5 a CSS3. V úvodu práce jsou popsána již existující řešení a jejich omezení ve vazbě na moderní trendy v HTML5. Další část je věnována analýze a návrhu vlastního řešení s ohledem na univerzálnost a možnost použití pro nejrůznější myslitelné účely. V závěru práce jsou popsány provedené výkonnostní testy a zdokumentovány dosažené výsledky.

**Klíčová slova**: node editor, HTML5, CSS3, bakalářská práce

**Abstract**

This bachelor thesis describes design and implementation of web application / component, which plays a role of so called node editor. Application uses an actual possibilities of HTML5 and CSS3. The beginning of a thesis covers existing solutions and their limitations related to modern trends in HTML5. Another part is devoted to analysis and design of the solution considering universality and possibility of using for any conceivable purpose. In the end of a theses, performance tests are described and achieved results are discussed.

**Key Words**: node editor, HTML5, CSS3, bachelor thesis

# Obsah

# Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| HTML | – | HyperText Markup Language |
| CSS | – | Cascading Style Sheets |
| LESS | – | Leaner CSS |
| JSON | – | JavaScript Object Notation |
| GUI | – | Graphical User Interface |
| DOM | – | Document Object Model |

# 1 Úvod

Obsahem této bakalářské práce je rozbor problematiky tzv. node editorů. Node editor je grafický nástroj, který používá tzv. nodů (uzlů) a jejich spojování pro přehlednou vizualizaci programového procesu. Uzly mohou mít vstupy, kam mohou vstupovat data z jiných uzlů, výstupy, kde jsou připravena data generovaná uzlem a obsah - zpravidla uživatelem nastavitelné hodnoty v rámci uzlu, která se používají pro výpočet výstupu. Uzly si můžeme představit jako programovatelné jednotky plnící zadanou úlohu na základě dat, která mají na vstupu, nebo která do uzlu vyplní uživatel. Tímto způsobem lze pomocí většího počtu propojených uzlů vytvářet poměrně složité struktury, které jsou ale díky tomuto způsobu zobrazování stále dobře čitelné a srozumitelné uživateli. Proto mají velký potenciál využití v aplikacích nejrůznějšího druhu a v současnosti rychle přibývá aplikací využívajících těchto principů.

V první části práce je zmapován současný stav využití principů node editoru a vyjmenováno několik příkladů použití node editoru v existujících aplikacích. Node editory v těchto aplikacích jsou určeny právě pro tuto konkrétní aplikaci a jejich použití v rámci jiné aplikace s odlišným účelem je i u open source řešení velmi omezené. Tyto editory totiž nejsou dostatečně konfigurovatelné, obhsaují pouze předdefinované typy uzlů, které jsou nutné pro potřeby daného programu. Ohnutí takového editoru pro vlastní potřeby tedy vyžaduje nemalé zásahy do samotného kódu programu. Často pak bývá pro vývojáře výhodnější vytvořit si vlastní řešení, které se více hodí pro daný účel. Takové řešení ale opět nebude snadno použitelné nikde jinde, než v dané aplikaci. Principy fungování editoru jsou však všude stejné a vývojář je nucen znovu implemetovat funkčnosti, které už dávno implementovalo mnoho jiných před ním.

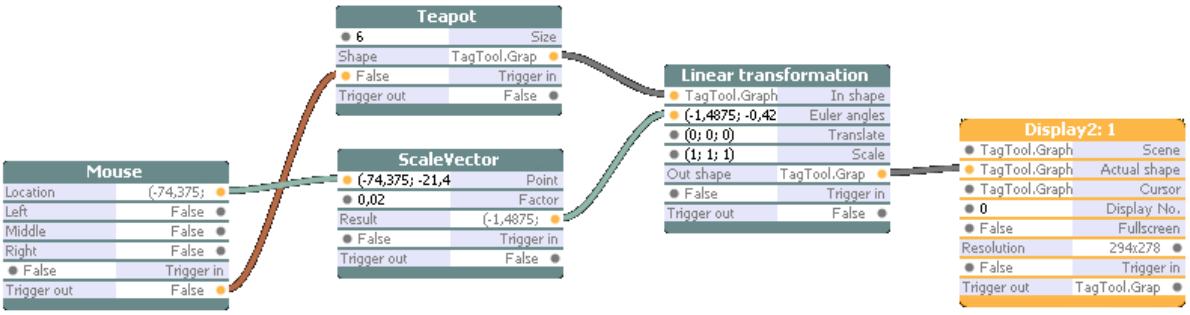
Cílem a hlavním obsahem této bakalářské práce je tedy návrh a implemetace univerzálního řešení pro HTML5 a javascript, které bude fungovat samostatně bez zastřešující aplikace a bude volně konfigurovatelné pro jakýkoli účel. Aplikace by měla nabízet všechny standardní funkce pro práci s uzly, které jsou dnes u node editorů běžné. Měla by klást důraz na příjemné uživatelské rozhraní a co největší a nejjednodušší konfigurovatelnost. Editor by měl umožňovat vytváření nových typů uzlů včetně jejich programovatelné části. Mělo by také být možné vytvořené konfigurace ukládat a načítat a případně nastavit uložené konfigurace pro načtení při startu programu. Editor by tak měl být snadno použitelný jako plugin pro jakoukoli další aplikaci, která by pro své fungování potřebovala tento typ rozhraní a vývojář by se nechtěl zbytečně zdržovat jeho implementací.

Poslední část práce se pak bude věnovat výkonnostnímu testování navrženého node editoru v nejpoužívanějších prohlížečích a jeho optimalizaci pro co nejlepší výsledky i při velkém množství zobrazených uzlů a složité aplikační logice.

# 2 Přehled existujících řešení

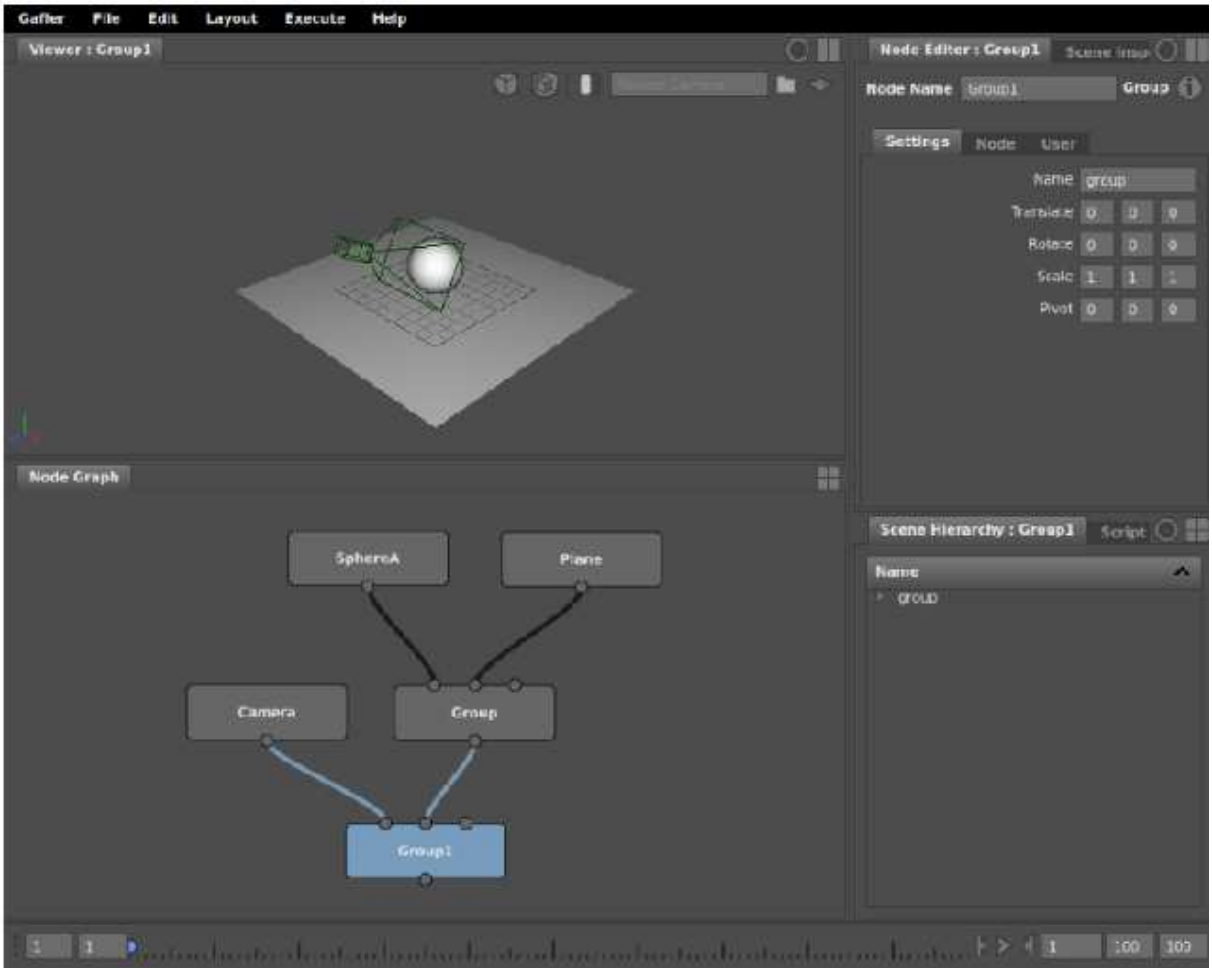
Node editory jsou užitečnou součástí mnoha stávajících aplikací z širokého sektra odvětví. Nejčastěji se lze s node editory setkat v aplikacích pro tvorbu grafiky a animací (Blender, Autodesk Maya LT...), dále pak jsou hojně používány při tvorbě her (SoundCue Editor, NodaCanvas). Stejné principy jsou ale použitelné i v různých jiných oblastech (zde za všechny Circuit Maker) Zde je několik příkladů reálného použití v praxi.

- Tagtool: Softwarový nástroj pro malování a tvorbu animací a vizuálů. Obsahuje open-source node editor Nodekit, který slouží k výraznému usnadnění práce s programem. Tento node editor byl vyvinut speciálně pro aplikaci Tagtool, avšak funguje jako modul a lze jej použít i v rámci vlastrního řeešní. Na obrázku č. 1 je příklad použití pro vykreslení čajové konvice, jejíž zobrazení lze rotovat tažením myší. Vlevo je uzel reprezentující myš, tedy vstup. Jeho vlastnost location je vstupem do uzlu ScaleVector, kde je přepočítána podle určitého indexu a dál se pošle do uzlu Linear Transformation, kde se provede samotná rotace konvice. Aby se prováděla rotace pouze při stisknutém tlačítku myši, je spojen výstup „Trigger out“ z myši s „Trigger in“ u čajové konvice. Uzel „Display2:1“ pak reprezentuje výsup, tedy samotné zobrazení konvice.



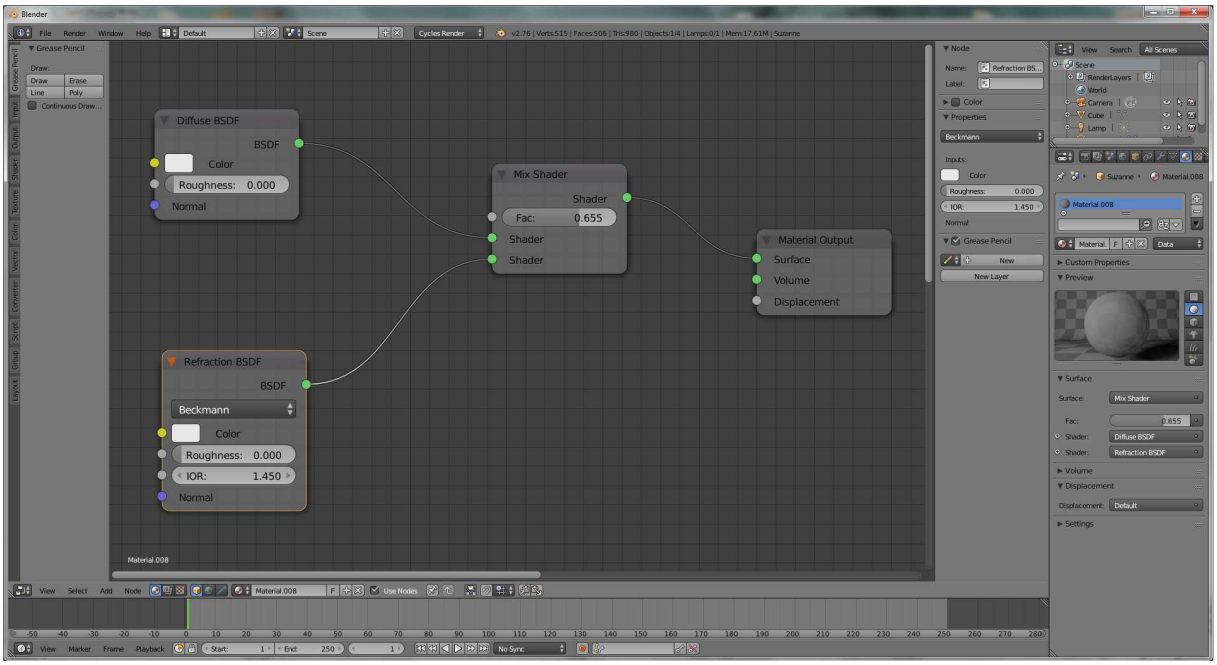

Obrázek 1: Nodekit

- Gaffer: Linuxový open-source nástroj pro tvorbu animací. Pro procedurální vyjádření scén používá síť uzlů, které se vytvářejí a upravují pomocí zabudovaného node editoru. Na obrázku máme jednoduchou scénu sestávající z uzlů reprezentujících kouli (SphereA) a rovinu (Plane). Těmto uzlům lze nastavit nejrůznější vlastnosti jako rozměry, barvu, texturu apod. Tyto objekty jsou spojeny do jedné scény pomocí uzlu Group. Dále je zde uzel reprezentující kameru (Camera), která snímá scénu pod určitým úhlem a z určité vzdálenosti. To je zajištěno spojením výstupu kamery a uzlu Group představujícího scénu do dalšího sjednocovacího uzlu Group1. Ten tedy reprezentuje celou scénu i s umístěnou kamerou.



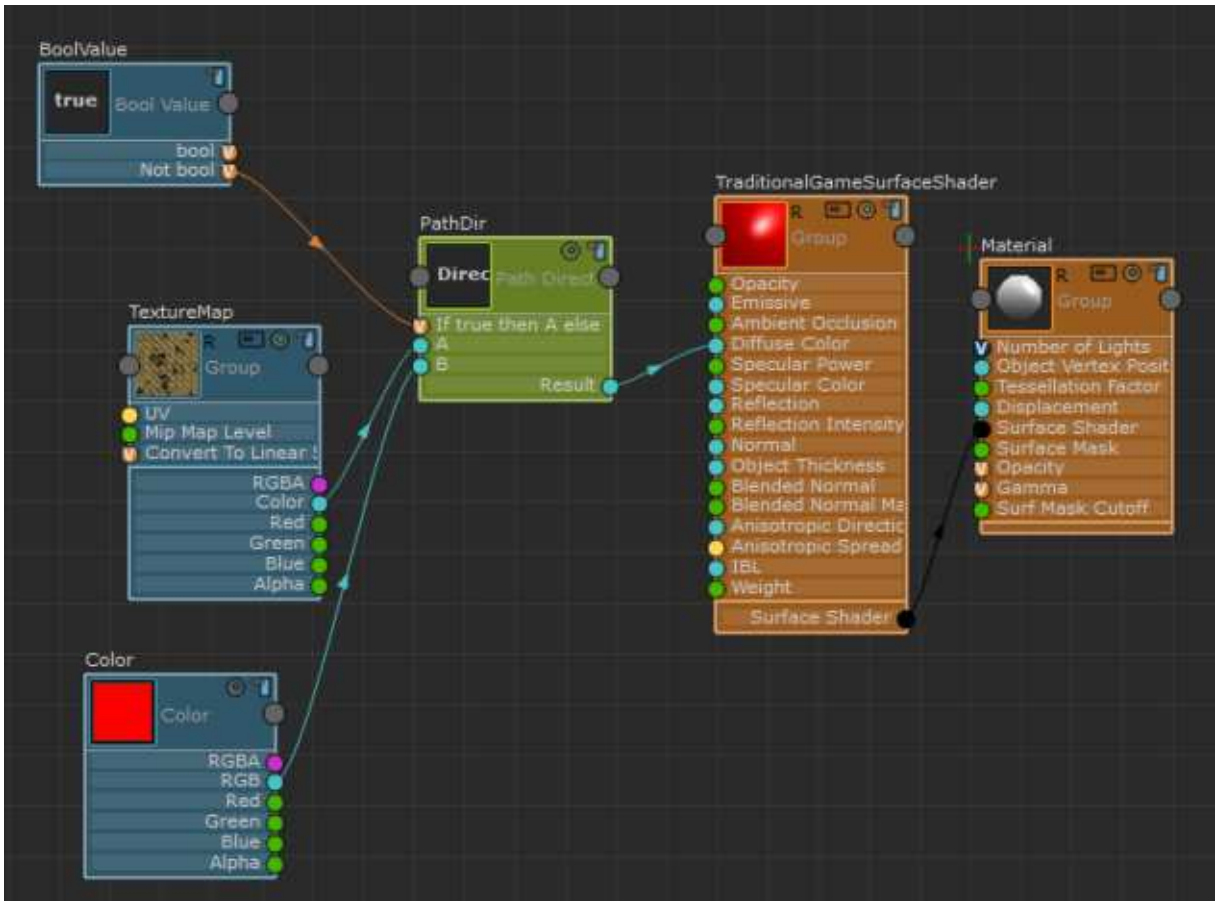

Obrázek 2: Gaffer

- Blender: Blender je softwarový nástroj pro modelování 3D grafiky za použití knihovny OpenGL. Slouží pro usnadnění práce s nejrůznějšími grafickými objekty. Blender obsahuje node editor od verze 2.42 a stal se poměrně oblíbeným a používaným. Od své první verze je neustále vylepšován a jsou přidávány nové typy nodů a různých funkčností a stal se poměrně komplexním a mocným nástrojem. Na obr. č. 3 je jednoduchá ukázka použití node editoru v Blenderu pro definici materiálu. Vlevo jsou dva materiálnové vstupy – jeden s matným povrchem, druhý s odrazivým. Lze u nich nastavit barvu, hrubost, index odrazivosti, atd. Jsou spojeny s nodem, který slouží pro kombinaci vlastností dvou vstupních materiálů podle zadaného poměru a ten je pak spojen s výstupem, kde lze zobrazit výsledek.



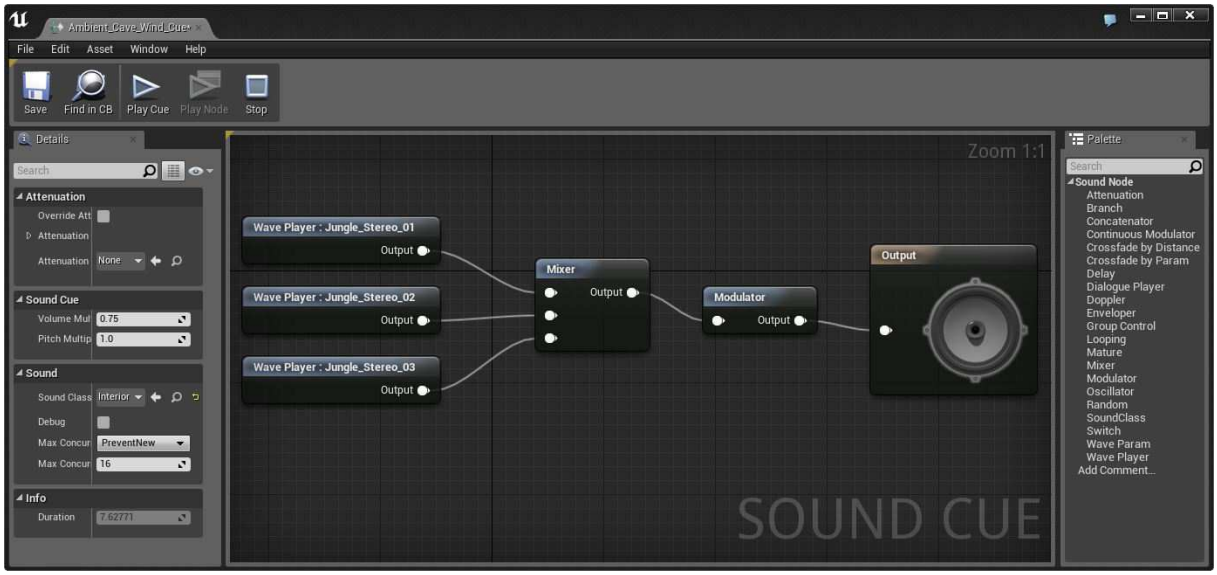

Obrázek 3: Blender

- Autodesk Maya LT: Je to nástroj pro 3D modelování a tvorbu animací. Je navržen pro tvorbu her pro PC i mobilní zařízení. Podobně jako u Blenderu obsahuje Maya node editor sloužící k výraznému usnadnění práce s programem. Na obrázku č. 4 je obdobný příklad použití node editoru, jako u Blenderu, kdy na výstupu je materiálový nod, jehož povrch je určen výsledkem vyhodnocení výstupů nodů nalevo od něj. Vlevo máme tři vstupy – texturu, barvu a booleovskou hodnotu. Všechny tyto vstupy jsou spojeny s podmínkovým nodem, který podle hodnoty booleovského nodu a podle zadaného výrazu pro vyhodnocování podmínky určí, zda na jeho výstupu bude textura, nebo červená barva. Na toto se následně aplikuje filtr a jeho výstup je pak výsledným povrchem materiálu.



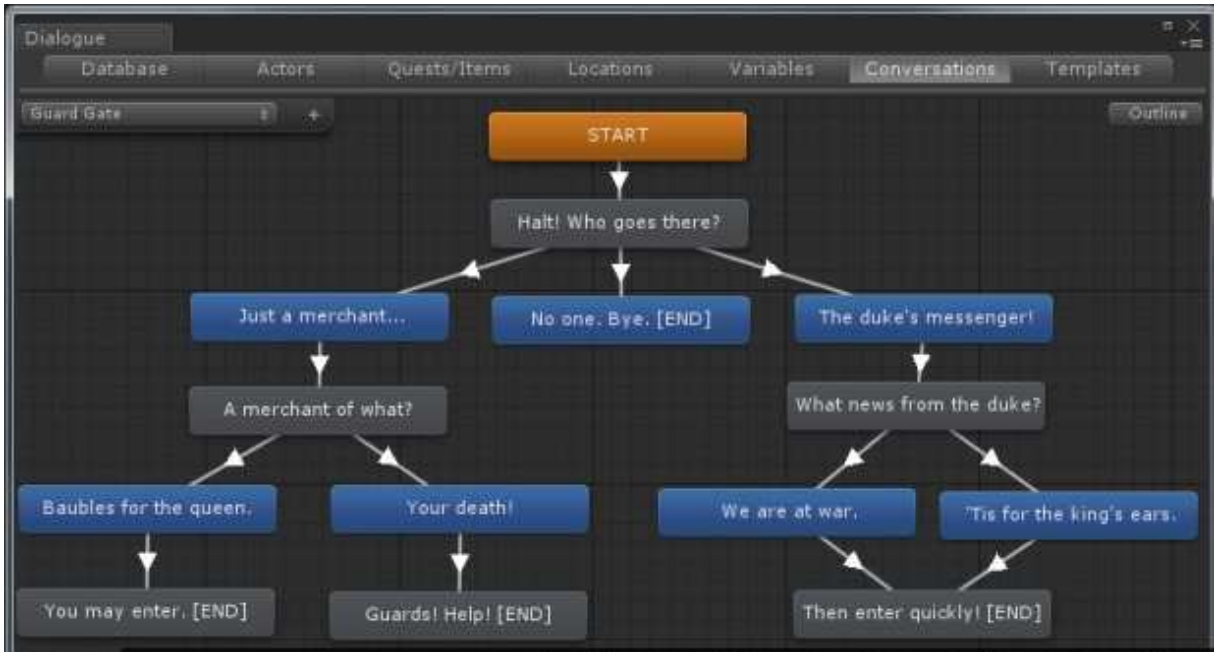

Obrázek 4: Maya LT

- Sound Cue Editor: Unreal herní engine používá pro zvuk tzn. sound cues. Pro jejich snadnější vytváření a úpravy slouží nástroj SoundCue editor. Na obrázku č. 5 je ukázka použití. Vstupem jsou tři uzly se zvukovými stopami. Ty jsou v určitém poměru smixovány v uzlu reprezentujícím mixer a výstup z něj je ještě následně upraven v modulátoru než konečně doputuje k výstupu (reproduktoru).



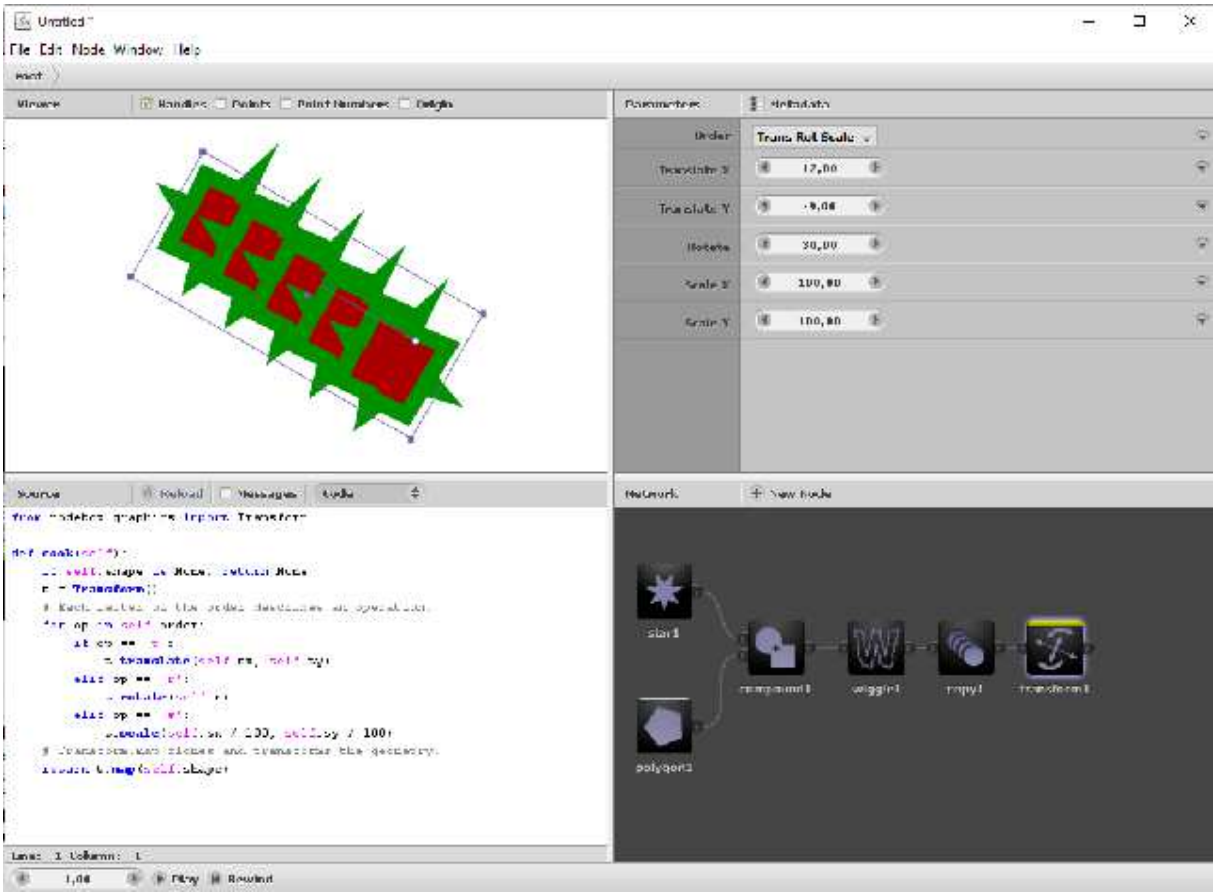

Obrázek 5: Sound Cue Editor

- NodeCanvas pro Unity: Unity je herní engine používaný pro tvorbu her pro nejrůznější zařízení. NodeCanvas je známé rozšíření pro tento engine a dá se použít např. pro tvorbu dialogových stromů, scénářů chování hry nebo organizaci stavů (běh, obrana, útočení, seskupování se apod.). Na obrázku č. 6 je příklad dialogového stromu, kde na vstupu je interakce hráče s osobou, která jej zastaví při vcházení do hradu. Podle znázorněného průběhu může konverzace skončit vpuštěním do hradu, přivoláním stráží nebo dobrovolným odchodem hráče.



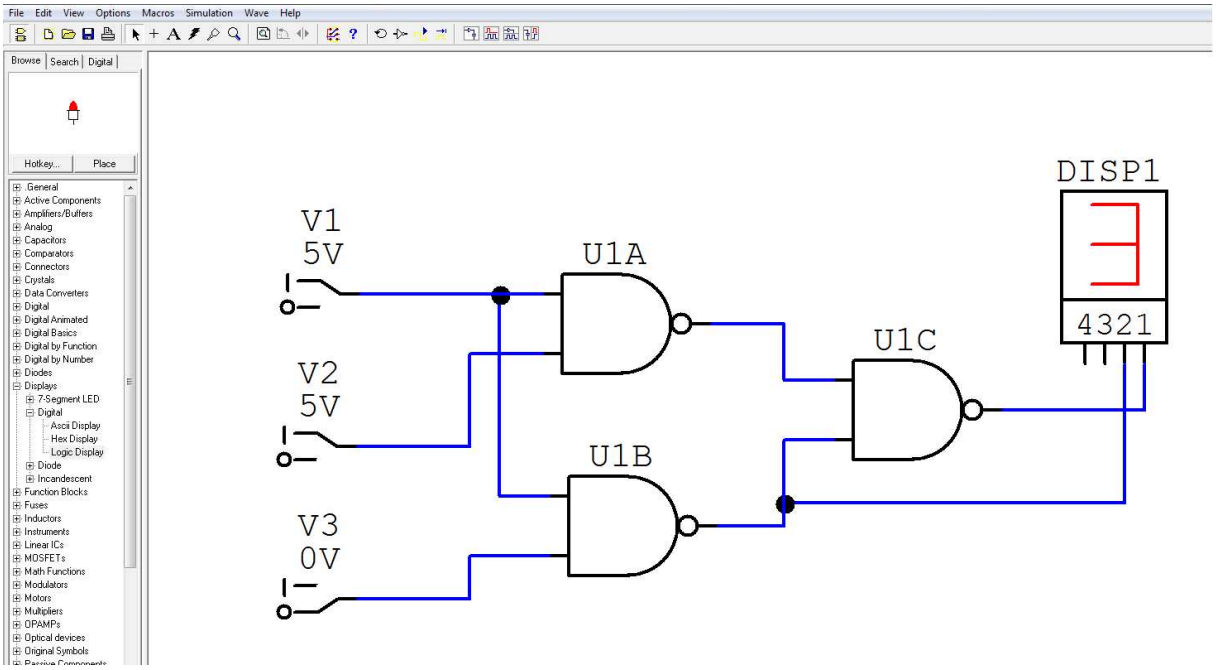

Obrázek 6: NodeCanvas

- NodeBox: Aplikace pro procedurální tvorbu vektorových obrázků. Pro vytvoření výsledného obrázku používá síť uzlů spravovaných zabudovaným node editorem. Na obrázku č. 7 v editoru vpravo dole vidíme na vstupu dva tvary – červenou hvězdu a červený polygon – které jsou napozicovány tak, aby se překrývaly. Tyto jsou následně skombinovány do jednoho tvaru pomocí uzlu coupound a v tomtéž uzlu je výslednému tvaru nastaven zelený okraj. Výstup z něj jde pak do uzlu wiggle, které provede náhodné zkroucení a deformaci tvaru podle zadaných parametrů. Pomocí dalšího uzlu v pořadí copy je obrázek pětkrát nakopírován a jednotlivým kopiím je nastavena vzdálenost na obou souřadnicových osách. Poslední uzel transform potom provede ještě rotaci a napozicování výsledku.



Obrázek 7: Nodebox

- Circuit maker: Jde o starý, ale stále velmi užitečný nástroj z devadesátých let pro simulaci elektrických obvodů, digitálních či logických. Třebaže zařazení tohoto programu mezi node editory je sporné, tak princip fungování je úplně stejný. Jako uzly zde figurují jednotlivé elektronické součástky, jako pojítka fungují elektrické vodiče. Vstupy mohou být například zdroje napětí, konečný výstup může být třeba spotřebič. Na obrázku č. 3 je ukázka jednoduchého logického obvodu. Na vstupu jsou jednotlivé spínací zdroje napětí generující napětí 5V (jednička) nebo 0V (nula). Prostřednictvím vodiče jsou spojeny určitým způsobem se třemi logickými hradly typu NAND a jejich výstup je napojen na digitální displej zobrazující výslednou hodnotu.




Obrázek 8: Circuit Maker

# 3 Analýza

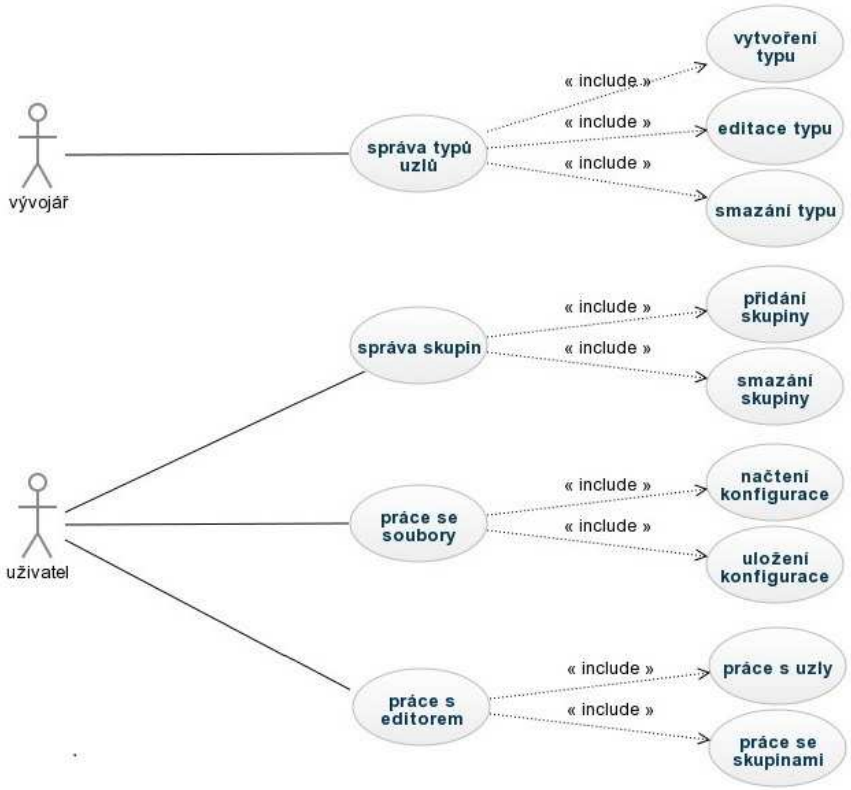
Program node editoru bude sloužit k definování uzlů (nodů) a jejich následné používání a propojování v aplikaci. Aplikace by měla být nezávislá a snadno použitelná jako plugin v rámci většího řešení. Grafické uživatelské rozhraní editoru bude implementovat všechny potřebné funkce pro práci s uzly (přemisťování, označování, shlukování do skupin, zoom apod.).

U uzlu bude možné definovat jeho vstupy, výstupy a obsah a u každého výstupu také kód, který bude vyhodnocovat obsah uzlu, posílat data na výstup a tím definovat jeho funkci. Pomocí programovatelné definice funkce uzlu tedy bude možné použít editor pro jakýkoli myslitelný účel. Vytvořená schémata uzlů pak bude možné ukládat a načítat do / ze souboru.

Princip node editoru je použitelný pro řešení široké palety nejrůznějších problémů, v současnosti ale chybí volně dostupný node editor, který by byl uviverzálně použitelný a bylo by možné ho konfigurovat na jakoukoli funkci. Typicky tedy bude tento projekt sloužit vývojáři aplikace, která potřebuje pro své fungování node editor a bude se chtít vyhnout vývoji vlastního řešení.

## 3.1 Funkční analýza

V této podkapitole zevrubně popíšeme aplikaci z uživatelského hlediska. Níže jsou popsány všechny případy užití a druhy uživatelů, kteří mohou s aplikací pracovat.




Obrázek 9: Diagram případů užití

### 3.1.1 Uživatelé

Node editor je možné spouštět ve dvou módech, které odpovídají typu uživatele, který s editorem pracuje. Prvním módem je "developer". V tomto módu je dostupný celý editor se všemi svými funkčnostmi a bude sloužit vývojáři pro připravení editoru a předdefinování typů uzlů pro své konkrétní účely. Svou předdefinovanou konfiguraci editoru si pak může vyexportovat do souboru. Pokud tento soubor uloží do dané složky v aplikaci s určeným názvem, bude se tato konfigurace načítat vždy při spuštění aplikace. Druhým módem je "user". Zde je skryt nástroj pro definici typů uzlů a zobrazeny jsou pouze typy uzlů předdefinované vývojářem bez možnosti jejich úpravy či smazání.

### 3.1.2 Správa typů uzlů

- vytvoření typu: Po kliku na příslušné tlačítko v menu se zobrazí nástroj pro vytvoření typu uzlu. V něm je možné nastavit název typu uzlu a dále přidávat položky uzlu.

  Vkládané položky mohou být následujícího typu:

  - vstup: datový vstup do uzlu, v aplikaci je pak možné do něj připojit výstup jiného uzlu
  - výstup: datový výstup z uzlu, v editoru je ho možné buď pouze zobrazit nebo připojit na vstup jiného uzlu
  - obsah: nastavitelná část uzlu, podle datového typu obsahu se pak v editoru zobrazí příslušná komponenta pro zadání hodnoty uživatelem

  Název typu uzlu je povinný údaj a musí být unikátní. U každé přidané položky (vstupu, výstupu i obsahu) bude možné nastavit název a datový typ, kde obojí je povinné a název položky musí být unikátní v rámci uzlu. V případě výstupu také bude možné nastavit funkci pro výpočet hodnoty výstupu na základě hodnoty ostatních položek v uzlu.

  Datové typy položky mohou být tyto:

  - int: 32-bitové celé číslo
  - float: 32-bitové desetinné číslo
  - boolean: booleovská hodnota true nebo false
  - string: libovolný řetězec s neomozenou délkou
  - select: výběr z definovaných hodnot
  - image: obrázek
  - color: barva

  V případě datového typu "select"se zobrazí i povinné pole pro výčet hodnot, ze kterých bude možné v rámci položky vybírat. U datového typu "image"se bude konkrétní obrázek zadávat

formou url - odkazu na web. U každé přidané položky budou pak v rámci konfiguračního nástroje zobrazeny možnosti "upravit"a "smazat". Po dokončení konfigurace typu uzlu a zvolení volby "uložit"se uloží nový typ uzlu a v menu se objeví nová položka odpovídající tomtuto typu.

- editace typu: U každého typu uzlu v menu bude dostupná volba pro jeho editaci. Po jejím zvolení se zobrazí totožný nástroj, jako v případě vytvoření a budou do něj automaticky vyplněny údaje editovaného uzlu.
- smazání typu: Volba smazání bude dostupná v menu u každého typu uzlu.

### 3.1.3 Správa skupin

- vytvoření skupiny: Po kliku na příslušené tlačítko v menu se zobrazí formulář pro zadání názvu skupiny. Po potvrzení formuláře je přidána do menu nová skupina.
- smazání skupiny: Volba smazání skupiny bude dostupná u každé vytvořené skupiny v menu. Smazáním skupiny dojde také ke smazání všech uzlů přiřazených do skupiny.

### 3.1.4 Práce se soubory

- uložení konfigurace: Po zvolení příslušné položky v menu se vygeneruje soubor s uloženou aktuální konfigurací editoru. Tato konfigurace bude obsahovat seznam všech definic typů uzlů, seznam definic skupin, seznam uzlů vložených do editoru a hodnot jejich obsahových a výstupních položek, seznam jejich spojení a zařazení do skupiny. Pokud je vygenerovaný soubor uložen pod určeným názvem do dané složky v aplikaci, je tato konfigurace použita při startu aplikace.
- načtení konfigurace: Po kliku na tlačítko v menu se zobrazí okno pro výběr souboru na disku. Po vybrání souboru bude načtena konfigurace editoru uložená v souboru.

### 3.1.5 Práce s editorem

- práce s uzly: Všechny dostupné typy uzlů jsou zobrazeny v menu a lze s nimi provádět následující operace:
  - vložení uzlu: Vložení do editoru proběhne skrze zvolení daného typu v menu. Ten se následně vykreslí do editoru a pojmenuje se názvem typu uzlu.
  - smazání uzlu: Uzel se odstraní z editoru, spolu s uzlem se automaticky vymažou i všechna spojení, která má s ostatními uzly.
  - přejmenování uzlu: Uzel je možné pojmenovat jakýmkoli názvem, není vyžadována unikátnost.

– spojení uzlů: Proběhne propojení výstupu jednoho uzlu se vstupem druhého uzlu. Pokud výstupní uzel generuje nějakou hodnotu, tato se přenese na vstupní uzel a je tam zobrazena. Dojde pak k přepočtení hodnot všech položek vstupního uzlu i všech dalších uzlů, které jsou spojeny s jeho výstupy. Před vytvořením spojení se budou kontrolovat datové typy vstupu a výstupu podle níže uvedené tabulky a dále se bude kontrolovat, zda spojením nedojde k vytvoření nežádoucího cyklu.

| datový typ vstupu | možné datové typy výstupu |
|---|---|
| int | int, boolean |
| float | int, float, boolean |
| string | všechny |
| boolean | boolean |
| select | všechny |
| image | image |
| color | color |

Tabulka 1: kompatibilita datových typů

- práce se skupinami: Všechny vytvořené skupiny jsou dostupné v menu a lze s nimi provádět následující operace:
  - označení uzlů ve skupině: Zvolením skupiny v menu dojde k označení všech uzlů v dané skupině. S těmito uzly je tam možné provádět hromadné operace přesouvání či smazání.
  - sbalení skupiny: U každé skupiny je dostupná volba pro sbalení skupiny. Zvolením této možnosti dojde k seskupení všech uzlů náležících do dané skupiny do jednoho skupinového uzlu s výpisem názvů všech zařazených uzlů. Všechna spojení, která existují mezi uzly uvnitř skupiny přestanou být zobrazována. Spojení, která vedou k uzlům mimo skupinu, budou pak zobrazena jako spojení se vstupem či výstupem ze skupinového uzlu. Se skupinovým uzlem pak lze provést operaci smazání. Tato volba způsobí smazání všech uzlů ve skupině, nedojde však k odstranění skupiny samotné.
  - rozbalení skupiny: Tato volba bude dostupná v menu a u sbaleného skupinového uzlu. Po zvolení se opět zobrazí všechny uzly ve skupině i se svými vnitřními spojeními a skupinový uzel je z editoru odstraněn.

## 3.2 Technická specifikace

- Architektura: Jedná se o offline javascriptovou aplikaci, tzn. k jejímu spuštění není potřeba webový server ani si ke svému fungování nestahuje žádné komponenty z externích zdrojů na internetu. V aplikaci je použita třívrstvá architektura, je tedy rozdělena na prezentačn vrtsvu, aplikační vrstvu a vrstvu přístupu k datům.

- Prezentační vrstva: Zajišťuje vykreslování veškerého obsahu editoru do prohlížeče a zprostředkovává interakci editoru s uživatelem. Tato interakce je realizována pomocí událostí. Pro vykreslení editoru se používá HTML se standardy HTML5 a kaskádové styly se standardy CSS3. Soubor s kaskádovými styly `editor_styles.css` je generován pomocí knihovny LESS, která kompiluje zdrojový soubor `editor_styles.less`. Logika prezentační vrstvy je implementována v souboru `editor.js`.

- Aplikační vrstva: Obsahuje bussiness objekty, které zapouzdřují data potřebná k fungování aplikace. Vrstva zprostředkovává komunikaci s vrstvou přístupu k datům. Je reprezentována těmito soubory:
  `editor_node.js`, `editor_node_contentItem.js`, `editor_node_group.js`, `editor_node_connection.js`.

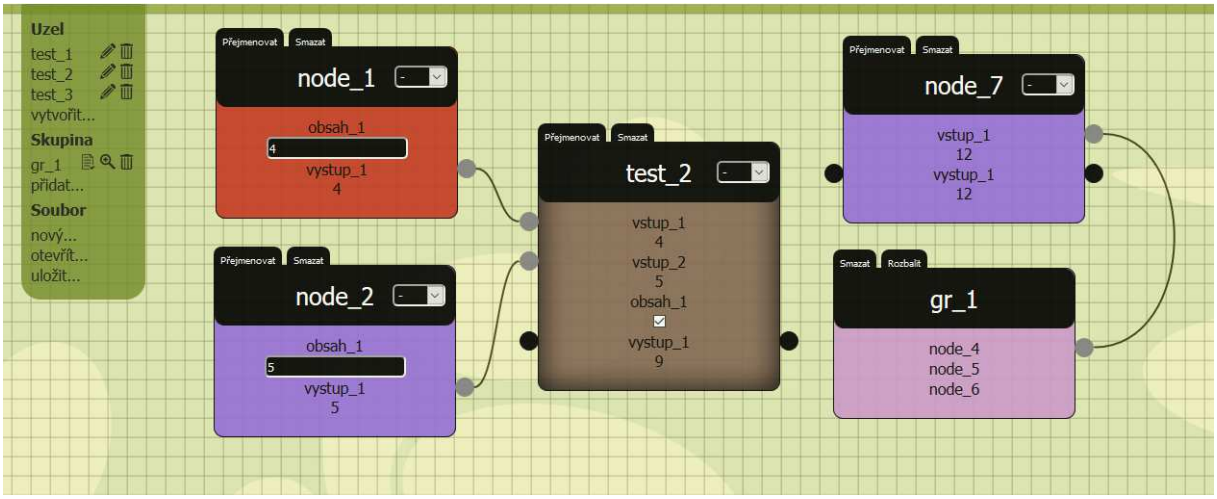
- Vrstva přístupu k datům: Tato vrstva zajišťuje veškerou práci s daty, která vstupují a vystupují za aplikace. Spravuje tedy export a import vytvořených konfigurací do souboru a načítání předpřipravené konfigurace při startu aplikace. Je reprezentována souborem `editor_data.js`. Práce s daty se také obejde bez serveru, pro práci s lokálními soubory je použito FileAPI, které je součástí specifikace HTML5.

- Uložení dat: Všechna data jsou ukládána a načítána ve formátu JSON. Tento formát byl vybrán proto, že je vhodný pro uložení větších objemů dat a jeho zpracování pomocí javascriptu je velice snadné. Pro serializaci ukládaných objektů je použita knihovna JSONfn. Je vybrána proto, že s pomocí této knihovny je možné serializovat i funkce náležící k objektu. Standardní knihovna JSON dostupná v prohlížeči toto neumí.

- Potřebný software: Pro spuštění aplikace stačí jakýkoli webový prohlížeč, který podporuje standardy HTML5 a CSS3 a má povolen javascript. Vzhledem k tomu, že se jedná o offline aplikaci, není třeba žádný webový server ani připojení k internetu.

## 3.3 Grafické uživatelské rozhraní

V této kapitole postupně projdeme všechny součásti editoru z hlediska grafického rozhraní.




Obrázek 10: Node editor

### 3.3.1 Menu

Ve výchozím stavu je menu sbaleno. Po kliku na některou ze tří položek se daná položka rozbalí. Menu se skládá ze tří částí - uzel, skupina a soubor:

- uzel: Je zde seznam vytvořených typů uzlů. Nový uzel do editoru se dá vložit kliknutím na položku v menu nebo přetažením položky do editoru. Tímto se spustí animace vytvoření uzlu. Vedle každého typu uzlu jsou ikony pro vymazání a editaci. Poslední položkou v menu uzel je tlačítko "vytvořit...", které otevře nástroj pro vytvoření typu uzlu.
- skupina: Je zde seznam vytvořených skupin. Kliknutím na skupinu se označí všechny uzly zařazené do skupiny. Vedle každé skupiny jsou ikony pro spuštění akcí zařazení do skupiny, rozbalení / sbalení skupiny a smazání skupiny. Akce zařazení do skupiny zařadí všechny označené uzly do dané skupiny. Akce sbalení skupiny spustí animaci na všechny uzly v dané skupině, které se "seběhnou"do jednoho místa, kde se zobrazí sbalený skupinový uzel (viz. Obrázek 10, uzel s názvem "gr1"). Místo je dáno aritmetickým průměrem pozic všech uzlů ve skupině. Akce rozbalení skupiny spustí opačnou animaci, uzly se rozeběhnou do svých původních pozic před sbalením. Poslední položkou je tlačítko "přidat...", které zobrazí formulář pro přidání skupiny.
- soubor: Tlačítko "nový..."po potvrzení vyčistí editor - spustí animaci zmizení na všechny uzly v konfiguraci. Tlačítka "otevřít..."a "uložit..."ukládají a načítají konfigurace editoru z / do souboru. Při načtení konfigurace ze souboru se nejprve aktualizuje obsah menu, pak se spustí animace objevení uzlu na všechny uzly v konfiguraci. Po dokončení této animace jsou vykreslena také všechna spojení.

### 3.3.2 Uzly

Uzly se zobrazují na pracovní ploše (viz. Obrázek102) a sestávají ze čtyř částí:

- titulek: Nachází se v horní části uzlu, zobrazuje se v něm název uzlu a také select-box pro výběr skupiny, pokud jsou nějaké skupiny dostupné. Uzel se dá za titulek "chytit"a kamkoli přetáhnout. Kliknutím na titulek se uzel označí.

- záhlaví: Obsahuje tlačítka pro přejmenování a smazání uzlu. Stisknutí tlačítka pro přejmenování způsobí zobrazení textboxu na místě názvu uzlu, jehož vyplněním a potvrzením enterem nebo kliknutím mimo textbox se uzel přejmenuje. Tlačítko pro smazání spustí animaci pro smazání uzlu.

- tělo: Podle definice vstupů, výstupu a obsahů se sem tyto vykreslí. U vstupů a výstupů se vykreslí název a displej pro zobrazení hodnoty, o obsahů se zobrazí komponenta pro nastavení hodnoty uživatelem příslušná datovému typu položky. V případě datových typů int, float a image se zobrazí textbox, kde se u vkládané (int nebo float) hodnoty kontroluje, zda se jedná o validní celé respektive desetinné číslo. U datového typu string se zobrazí textarea, u datového typu boolean se zobrazí checkbox, datový typ select reprezentuje selectbox a u výběru barvy se zobrazí komponenta "tinyColorPicker".

- konektory: Zobrazují se stranách uzlu u všech vstupů a výstupů a slouží ke spojení se vstupy či výstupy jiných uzlů. Konektory položek, které nemají žádné spojení se zobrazují párově po obou stranách uzlu. Konektory položek s aktivním spojením jsou barevně označeny a zobrazí se pouze na té straně uzlu, kam vede spojovací křivka. Pravidla pro vykreslování spojovací křivky jsou popsána níže.

### 3.3.3 Skupinové uzly

Zobrazují se na pracovní ploše po provedení akce sbalení skupiny. (viz. Obrázek 10, uzel "gr1") Sestávají ze stejných částí jako normální uzly, avšak mají rozdílný obsah:

- titulek: Zobrazuje název celé skupiny. Uzel se dá kliknutím na titulek označit nebo chytit a přesunout.

- záhlaví: Obsahuje tlačítko pro smazání uzlu a rozbalení skupiny. Stisk tlačítka spustí příslušnou akci.

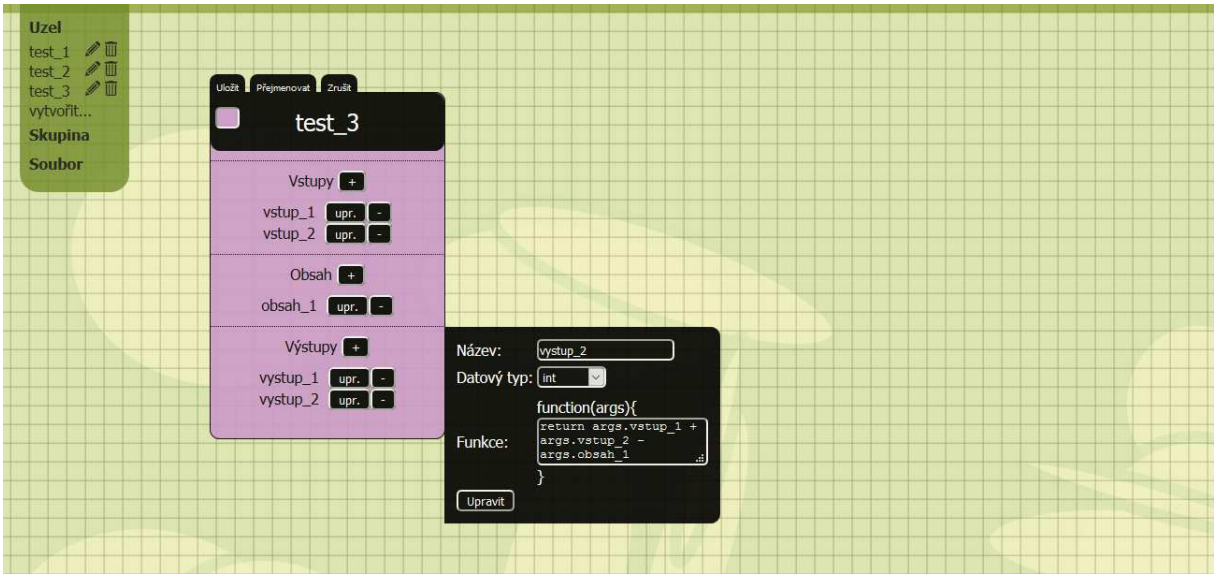
- tělo: Obsahuje pouze výpis názvů uzlů, které jsou ve skupině zařazeny.

- konektory: Zobrazí se pouze v případě, že některý uzel ve skupině je spojen s uzlem mimo skupinu. V takovém případě do skupinového konektoru vede spojení z daného uzlu mimo skupinu. Konektor je zde pouze kvůli zobrazení, nelze na něj navazovat další spojení na uzly uvnitř skupiny. Na to je potřeba skupinu rozbalit.

### 3.3.4 Nástroj pro vytváření typů

Zobrazuje se na pracovní ploše (viz. Obrázek 11) podobně jako uzly a skládá se také ze stejných součástí vyjma konektorů:

- titulek: Zobrazuje název typu. Uzel se dá se kliknutím na titulek označit nebo chytit a přesunout.
- výběr barvy: Zobrazí komponentu pro vybrání barvy. Tuto barvu pak budou mít všechny uzly tohoto typu vložené do editoru.
- záhlaví: Obsahuje tlačítka uložit, přejmenovat a zrušit. Tlačítko přejmenovat funguje stejně jako u uzlu a způsobí změnu názvu typu. Ostatní dvě tlačítka způsobí spuštění animace pro zmizení nástroje a uloží nebo zahodí provedené změny. V případě uložení nového typu se ještě spustí animace objevení nového typu v menu.
- tělo: Je rozděleno na tři části, ve kterých je možné libovolně přidávat, mazat a editovat vstupy, výstupy nebo obsah uzlu. Po stisknutí tlačítka pro přidání nebo úpravu vstupu, výstupu nebo obsahu vyjede po pravé straně nástroje formulář pro vyplnění či úpravu náležitostí položky. Těmi jsou její název, datový typ a v případě výstupu také vyhodnocovací javascriptový výraz / funkce. Po vyplnění a stisknutí tlačítka upravit nebo přidat formulář opět zajede, uloží se změny a v případě přidávání nové položky se také spustí animace objevení nové položky v těle nástroje. Po stisku tlačítka pro odstranění položky se spustí animace zmizení položky.




Obrázek 11: Nástroj pro vytváření typů uzlů

### 3.3.5 Spojení

Spojení existují vždy mezi konektory vstupu a výstupu dvou uzlů. Vykreslují se do canvasu a jsou realizovány tažením myši od jednoho konektoru ke druhému. Poté se zkontroluje validita spojení (viz. 3.1.5) a pokud vše projde, vykreslí se do canvasu mezi oběma konektory kubická Bézierova křivka (se dvěma kontrolními body) podle určených pravidel. Křivka se vykresluje už během tažení myši, jen jeden konektor je nahrazen aktuální pozicí myši. Konektory spojené křivkou se barevně označí a odpovídající párové konektory na stranách uzlů, které nespojuje křivka, zmizí. Spojením dojde k nastavení hodnoty vstupní položky na hodnotu výstupní položky a jejím zobrazení na displeji. Následně se přepočítají a zobrazí hodnoty vstupního uzlu a všech uzlů navázaných na jeho výstupy. Z jednoho výstupního konektoru lze vést libovolný počet spojení. Do jednoho vstupního konektoru ale smí vstupovat pouze jedno spojení. Pravidla pro vykreslování Bézierovy křivky mezi konektory jsou následující:

1. uzly neleží pod sebou (viz. Obrázek 10, uzly "node_1"s "test_2"nebo uzly "node_2"s "test_2"): Zjistí se to tak, že intervaly určené krajními body na ose x u obou uzlů nemají průnik. Horizontání souřadnice počátečního a koncového bodu $P[p_x, p_y]$ a $K[k_x, k_y]$ jsou dva k sobě přivrácené krajní body z obou intervalů. U jednoho z uzlů bude tedy spojen konektor na pravé straně s konektorem na levé straně druhého. Vertikální souřadnice obou bodů jsou vertikální souřadnice obou spojovaných konektorů (u pravého i levého jsou stejné). Souřadnice dvou pomocných bodů $A[a_x, a_y]$ a $B[_x, _y]$ pro vykreslení Bézierovy křivky se pak vypočtou jako:
   $a_x = (p_x + k_x) / 2$
   $a_y = p_y$
   $b_x = (p_x + k_x) / 2$
   $b_y = k_y$

2. uzly leží pod sebou (uzly "node_7"s "gr1"): Intervaly určené krajními body obou uzlů na ose x mají průnik. Tentokrát budou tedy spojeny buď oba pravé nebo oba levé konektory - o tom rozhodne vzájemná pozice vertikální os obou uzlů. Pokud vertikální osa vstupního uzlu je nalevo od vertikální osy výstupního uzlu, spojeny budou levé konektory. V opačném případě budou spojeny pravé konektory. Souřadnice pomocných bodů křivky $A[a_x, a_y]$ a $B[b_x, b_y]$ pak leží na ose y ve výšce obou konektorů a na ose y na půl cesty mezi oběma uzly. Výpočet vypadá následovně:

   (a) spojeny jsou levé konektory obou uzlů: Pro výpočet horizontální souřadnice použijeme bod $L[l_x, l_y]$, což jsou jsou souřadnice toho z obou konektorů, který je více vlevo. $P[p_x, p_y]$ a $K[k_x, k_y]$ jsou souřadnice spojovaných konektorů, tedy počáteční a koncový bod křivky. Pomocné body křivky mají pak souřadnice na ose y stejné, jako počáteční a koncový bod a na ose x leží 100 px nalevo od bodu $L$.
   $a_x = l_x - 100$

$a_y = p_y$
$b_x = l_x$ - 100
$b_y = k_y$

(b) spojeny jsou pravé konektory obou uzlů: Analogicky k prvnímu případu se použije bod `R[`$r_x$`,`$r_y$`]`, který představuje souřadnice konektoru ležícího více vpravo a pomocné body budou na ose x ve vzdálenosti 100px napravo od bode `R`. Výpočet bude vypadat takto.
$a_x = r_x$ + 100
$a_y = p_y$
$b_x = r_x$ + 100
$b_y = k_y$

### 3.3.6 Pracovní plocha

Pracovní plocha je místo, kde se zobrazují jednotlivé uzly, skupiny a spojení mezi nimi. Na pozadí pracovní plochy se zobrazuje mřížka. Aplikace reaguje na změnu velikosti okna prohlížeče nebo rozměrů dokumentu a automaticky přepočítává velikost pracovní plochy. Na pracovní ploše lze provádět tyto operace:

- označování: Označovat je možné skupiny, skupinové uzly, skupiny nebo spojení. Označené skupiny, uzly a skupinové uzly lze pak přesouvat po pracovní ploše nebo smazat stisknutím klávesy delete. Označená spojení lze pouze smazat. Označovat uzly a skupinové uzly lze kliknutím na příslušný objekt, skupinu je možné označit kliknutím na příslušnou položku v menu. V případě uzlů, skupinových uzlů a skupin je detekce kliknutí jednoduchá pomocí jquery události. V případě spojení je použita funkce `isPointInPath()` objektu `Path2D`, ve kterém jsou uloženy všechny aktuálně vykreslené Bézierovy křivky. Pro detekci kliknutí na danou křivku je určená tolerance 5px nad i pod křivkou, aby nebylo potřeba se kliknutím trefit přesně do křivky. Kliknutím kamkoli do editoru mimo nějaký objekt se veškerá označení zruší.

  Hromadné označování je možné dvěma způsoby. První způsob je pomocí podržení klávesy shift a postupném kliknutí na všechny objekty, které chce uživatel označit. Takto lze hromadně označovat všechny objekty. Druhý způsob je přetažením myši přes určitou oblast. Tím se vytvoří a vykreslí do canvasu výběrový obdélník a všechny objekty uvnitř obdélníku se označí. Pro označení stačí, aby se výběrový obdélník objektu pouze dotknul. Není potřeba, aby v obedélníku byl objekt celý. Metoda detekce označení pomocí výběrového obdélníku se liší u html objektů (uzel, skupinový uzel) a objektů v canvasu (Bézierových křivek):

– detekce označení uzlu a skupinového uzlu: Projdou se všechny uzly a skupinové uzly a provede se na nich následující test. Nejprve je třeba zjistit krajní souřadnice na obou osách kontrolovaného uzlu $a_{x1}$, $a_{y1}$, $a_{x2}$, $a_{y2}$. Pak je třeba zjistit krajní souřadnice na obou osách výběrového obdélníku $b_{x1}$, $b_{y1}$, $b_{x2}$, $b_{y2}$. Kontrolovaný uzel je pak označen, pokud je splněn následující logický výraz S, kde V je logická funkce "je větší než"a M je logická funkce "je menší než".

$$S = \overline{V(a_{x1}, b_{x2}) \vee M(a_{x2}, b_{x1}) \vee V(a_{y2}, b_{y1}) \vee M(a_{y1}, b_{y2})}$$

– detekce spojení: Zde je třeba si vypomoci výpočtem pomocných bodů na Bézierově křivce. Bylo by sice možné pomocí matematického vyjádření křivky kontrolovat všechny body ležící na křivce, ale to není z výkonnostních důvodů příliš vhodné. Je lepší si vypočíst dostatečný počet pomocných bodů na křivce a u těch potom během tažení myši prověřovat, zda některý z nich není obsažen ve výběrovém obdélníku. Pokud ano, dané spojení se barevně označí. Pokud je nějaké spojení označeno a žádný pomocný bod se nenachází ve výběrovém obdélníku, označení se zruší. Pokud souřadnice bodu na přímce jsou $a_x$, $a_y$ a krajní souřadnice výběrového obdélníku jsou $b_{x1}$, $b_{y1}$, $b_{x2}$, $b_{y2}$, V je logická funkce "je větší než"a M je logická funkce "je menší než", logický výraz S pro vyhodnocení, zda bod leží uvnitř výběrového obdélníku vypadá takto:

$$S = V(a_x, b_{x1}) \wedge M(a_x, b_{x2}) \wedge V(a_y, b_{y1}) \wedge M(a_y, b_{y2})$$

Pro výpočet bodů na křivce je použit algoritmus de Casteljau. Do algoritmu vstpuje parametr `t` nabývající hodnot od 0 do 1, který určuje, na které části křivky se vypočtou souřadnice na ní ležícího bodu. Pokud je například `t=0,5`, pak se bod vypočte v polovině křivky, pokud je `t=0,25` pak bude ve čtvrtině křivky apod. Algoritmus pracuje tak, že projde začáteční bod křivky, kontrolní body a konečný bod v tomto pořadí, na úsečkách mezi všemi sousedními body $P_0(p_{0x}, p_{0y})$ a $P_1(p_{1x}, p_{1y})$ vypočte polohu bodu $P(p_x, p_y)$ vzhledem k parametru `t`:

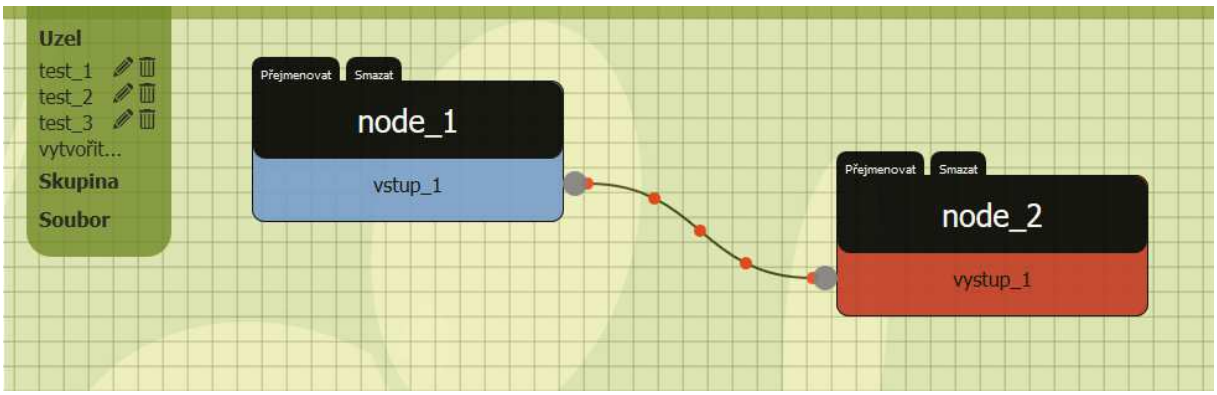
$$p_x = (1 - t)p_{0x} + tp_{1x}$$
$$p_y = (1 - t)p_{0y} + tp_{1y}$$

Bodů takto vypočtených je pak o jeden méně než původních bodů. Tento postup se opakuje tak dlouho, dokud nezbude jediný bod a ten je pak výsledkem. Algoritmus je tedy použitelný na jakoukoli Bézierovu křivku bez ohledu na počet kontrolních bodů. V ukázce kódu níže je javascriptová funkce, která pomocí tohoto algoritmu počítá body na křivce. Tyto body jsou pak použity pro označování spojení pomocí výběrového obdélníku. Do funkce vstupuje parametr `points`, tento obsahuje objekt se

seřazenými body definujicími křivku. Druhý parametr `computedPointsCount` určuje počet bodů, které mají být vypočteny. V aplikaci se u každé křivky počítá 5 bodů včetně počátečního a koncového bodu křivky (viz. Obrázek 12).

```
var DeCasteljau = function (points, computedPointsCount)
{
    var pointsToReturn = [];
    for (var i = 0; i < computedPointsCount; i++)
    {
        var pointsReduced = [];
        var pointsOrig = points;
        var t = i / (computedPointsCount - 1);
        while (pointsOrig.length > 1)
        {
            pointsReduced = [];
            for (var j = 0; j < pointsOrig.length - 1; j++)
            {
                pointsReduced.push({
                    x: (1 - t) * pointsOrig[j].x + t * pointsOrig[j + 1].x,
                    y: (1 - t) * pointsOrig[j].y + t * pointsOrig[j + 1].y
                });
            }
            pointsOrig = pointsReduced;
        }
        pointsToReturn.push(pointsOrig[0]);
    }
    return pointsToReturn;
}
```

Ukázka kódu 1: algoritmus de Casteljau



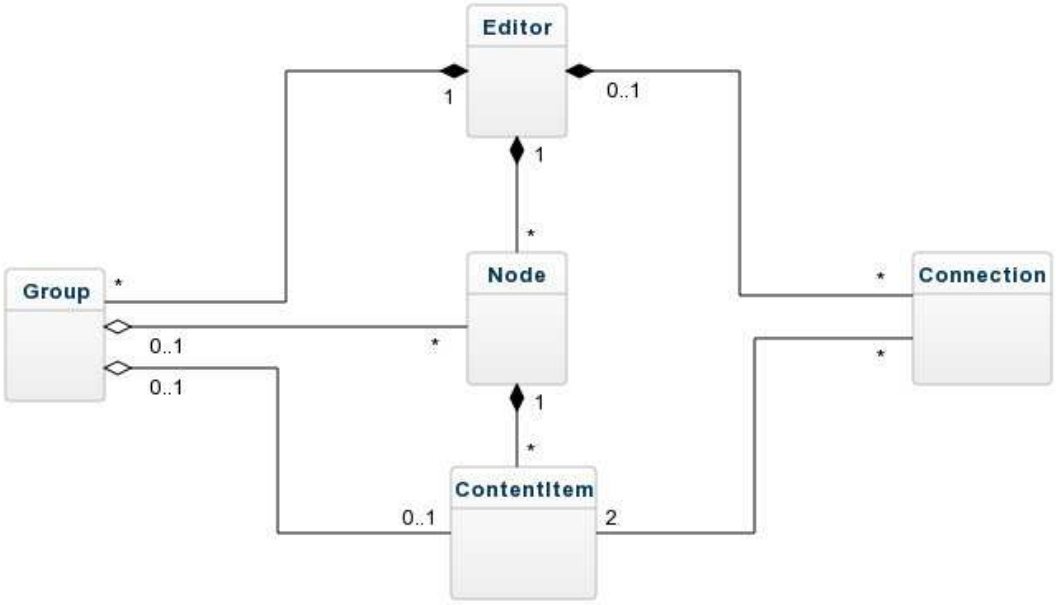

Obrázek 12: Body na Bézierově křivce vypočtené pomocí algoritmu de Casteljau

- zoom: Způsobí oddálení či přiblížení všech objektů na pracovní ploše. Provádí se kolečkem myši. Při otáčení kolečkem směrem od sebe se pracovní plocha oddálí, při otáčení směrem k

sobě se přibližuje. K tomu slouží globální proměnná ukládající koeficient, kterým se násobí CSS vlastnosti objektů, slouží pro výpočet křivek na canvasu, vstupuje jako parametr do nejrůznějších animací a operací s objekty tak, aby se vytvořil dojem oddálení či přiblížení. Koeficient nabývá hodnot od 0,4 do 1. Výchozí nastavení je 1, tedy přibližovat z výchozího nastavení není možné, lze pouze oddalovat.

## 3.4 Doménový model

### 3.4.1 Třídní diagram




Obrázek 13: Třídní diagram

### 3.4.2 Popis tříd

- Editor: Toto je hlavní objekt aplikace, děje se tady většina aplikační logiky. Uchovává si reference na všechny uzly, spojení a skupiny, které jsou zobrazeny v editoru. Závisí na něm všechny ostatní objekty, proto kompozice od skupiny, uzlu i spojení. Nejdůležitějšími funkcemi objektu jsou tyto:
  - Create(editor_settings): Tato funkce vykreslí celý editor do prohlížeče, vytvoří všechny potřebné datové struktury a spustí všechny další funkce pro definování obsluhy nejrůznějších událostí, které v editoru vznikají. Objekt editor_settings obsahuje všechna nastavení editoru: V této chvíli je dostupné pouze nastavení elemid. To definuje id objektu, do kterého se má vykreslit editor. V případě, že není vyplněno, editor se nevykreslí.

  - FillMenu(): Naplní menu předdefinovanými daty.
  - PrepareNodeRefreshing(): Definuje obsluhu událostí, které mají způsobit přepočet hodnot dotčených uzlů.
  - PrepareMenu(): Připravuje události, které nastávají v menu - přidávání uzlů, spouštění nástroje pro tvorbu typů, vybírání, sbalování a rozbalování skupin, ukládání a načítání dat ze souboru.
  - PrepareNodeSelecting(), PrepareGroupSelecting(), PrepareConnectionSelecting(): Definují funkčnosti umožňující označování, přemisťování a hromadné mazání uzlů a skupinových uzlů a označování a mazání spojení.
  - PrepareNodeConnecting(): Ošetřuje spojování uzlů
  - PrepareZoomFunction(): Připravuje funkci zoom.

- Node: Objekt zapouzdřuje uzel. Uzel může a nemusí náležet do skupiny, proto agregace s třídou Group. Může náležet nejvíce do jedné skupiny. Každý uzel musí mít přiřazen jeden objekt editoru, jinak nemá význam, proto kompozice s objektem Editor. Obsahuje také reference na všechny své položky (vstupy, výstupy, obsah). Nejdůležitějšími funkcemi jsou tyto:
  - GetHtml(): Vykreslí uzel do editoru.
  - AnimateIn(callback), AnimateOut(callback): Spustí sekvenci animací pro objevení / zmizení uzlu. Parametr callback je funkce, která se zavolá po dokončení animací.
  - Select(), Unselect(): Způsobí označení / zrušení označení daného uzlu.
  - RefreshValues(): Přepočte všechny hodnoty vstupů a výstupů v uzlu.

- Group: Objekt zapouzdřující informace o skupině. může obsahovat 0 až n uzlů. Agregace s třídou ContentItem je zde proto, že pokud je skupina minimalizovaná, může také podobně jako uzel (třída Node) obsahovat vstupy a výstupy, které se zobrazí ve skupinovém uzlu. Nejdůležitější funkce jsou:
  - GetHtml(): V případě, že skupina je sbalená, vykreslí do editoru skupinový uzel.
  - Minimize(), Maximize(): Sbalí / rozbalí všechny uzlu do / ze skupinového uzlu.
  - Select(), Unselect(): Označí / zruší označení u všech uzlů ve skupině.
  - AnimateIn(callback), AnimateOut(callback): Spustí sekvenci animací sbalení / rozbalení skupiny do / z skupinového uzlu. Parametr callback může obsahovat funkci, která se spustí po dokončení animací.
  - Move(diffX, diffY): Pokud je skupina minimalizovaná, přesune tato funkce skupinový uzel o diffX pixelů směrěm doprava a o diffY pixelů směrem dolů.
  - AddNode(node), RemoveNode(node): Přidá / odebere uzel ze skupiny.

- Connection: Zapouzdřuje data jednoho spojení. Spojení sestává z reference na dva objekty typu ContentItem. Nejdůležitější funkce objektu jsou:
  - Draw(): Tato funkce umí vykreslit na základě vzájemné polohy spojených uzlů spojení do canvasu.
  - DrawTemp(): Používá se během procesu spojování uzlů v aplikaci, kdy uživatel přetahuje spojení od jednoho uzlu k druhému. Už tady se vykresluje křivka, kdy druhým koncový bod křivky určují souřadnice myši.
  - DeCasteljau(): Vypočte pomocné body na přímce, které se pak využívají pro označení / zrušení označení spojení pomocí výběrového obdélníku (viz. Ukázka kódu 1 a Obrázek 12).
- ContentItem: Objekt zapouzdřující informace o položce (vstupu, výstupu nebo obsahu) uzlu. Pokud se jedná o vstup nebo výstup, může také uchovávat reference na spojení s jinými objekty typu ContentItem. Může se jednat o položku uzlu nebo skupinového uzlu. Nejdůležitější funkce objektu jsou tyto:
  - GetHtml(): Vykreslí danou položku v závislosti na jejím typu (vstup, výstup, obsah) a datovém typu (int, float, image...) do uzlu / skupinového uzlu. Je volána z funkce GetHtml() uzlu či skupinového uzlu.
  - RefreshValue(): Přepočte hodnotu položky. Pokud se jedná o vstup spojený s výstupem z jiného uzlu, načte si hodnotu z tohoto výstupu. Pokud se jedná o výstup, spustí vyhodnocovací funkci, přepočte svou vlastní hodnotu a pokud má spojení na další uzly, spustí na těchto uzlech funkci RefreshValues(), aby byla reflektována změněná hodnota výstupu.
  - Connect(connection), Disconnect(itemToDisconnect): Vytvoří / zruší spojení s položkou jiného uzlu.
  - CheckConnectedSide(): Podle spojení, které položka má na jiné uzly se zjistí, z které strany do položky vstupují. Pokud vstupují všechny pouze z jedné strany, konektor na druhé straně se skryje.

# 4 Testování

Bylo vytvořeno několik experimentů za účelem otestování výkonu aplikace a v případě neuspokojivých výsledků experimentů bylo provedeno několik úprav. Aplikace byla testována s tímto vybavením:
procesor: InterCore i7 2.93GHz
grafická karta: NVIDIA GeForce GT 330
operační systém: Windows 10
prohlížeče: Firefox 45.0.1, Google Chrome 45.0.2454.93 m, Microsoft Edge 25.10586.0.0

## 4.1 Experimenty

### 4.1.1 Překreslování canvasu

Za účelem zjištění rychlosti GUI byl vytvořen experiment. Je vytvořen speciální typ uzlu `test_1`, který má jeden vstup a jeden výstup. Na vstupu přijímá celé číslo a na výstupu k tomuto číslo přičte 1. Druhý speciální typ `test_2` nemá žádný vstup, v obsahu má nastavitelné celé číslo a to také emituje na výstup. Vygeneruje a vykreslí se do canvasu větší počet uzlů typu `test_1` s náhodnou pozicí a jeden uzel typu `test_2`. Tyto uzly se pak mezi sebou pospojují tak jak jdou za sebou, na začátku je uzel typu `test_2`. Testuje se rychlost překreslení canvasu, tedy všech spojení mezi uzly v závislosi na jejich vzájemné poloze. Testuje se tak, že je proveden určitý počet překreslení (10 - 100), změřen celkový čas a vypočítán průměrný čas na jedno překreslení. Tabulka výsledků je zde:

| počet spojení | rychlost Firefox (ms) | rychlost Chrome (ms) | rychlost Edge (ms) |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 5 | 4 | 3 | 8 |
| 10 | 8 | 5 | 16 |
| 20 | 15 | 13 | 40 |
| 100 | 73 | 53 | 423 |
| 500 | 380 | 270 | 9141 |

Tabulka 2: Měření rychlosti překreslení canvasu

Jak je vidět, rychlost překreslování není v žádném z prohlížečů dostatečná. Překreslování canvasu se provádí velmi často během manilupace s uzly v editoru - přesouvání, mazání, označování pomocí výběrového obdélníku apod. Bylo tedy třeba provést optimalizace. Jako největší brzda se ukázala být jakákoliv práce s DOM během vykreslování - manipulace s HTML objekty, čtení vlastností jako pozice, šířka, výška apod. Bylo tedy proveden několik úprav tak, aby se toto nemuselo dělat během vykreslování a maximum těchto infomací se předpočítávalo dopředu. Tabulka výsledků testování po provedených úpravách je zde:

| počet spojení | rychlost Firefox (ms) | rychlost Chrome (ms) | rychlost Edge (ms) |
|---|---|---|---|
| 1 | 0,5 | 0,3 | 1 |
| 5 | 0,9 | 0,8 | 3 |
| 10 | 1,8 | 1,5 | 6 |
| 20 | 3,5 | 3 | 12 |
| 100 | 19 | 14 | 63 |
| 500 | 94 | 67 | 318 |

Tabulka 3: Měření rychlosti překreslení canvasu po úpravách

Z výsledků měření rychlosti po úpravách vidíme, že došlo k výraznému zlepšení výkonu u všech prohlížečů. I v případě 100 vykreslených propojených uzlů se dal editor celkem bez problémů používat, animace byly plynulé. Při 500 propojených uzlech bylo také možné pracovat, ale GUI už bylo dost pomalé a reagovalo se zpožděním, nicméně v porovnání se stavem před úpravami, kdy bylo GUI už při 100 uzlech takžka nepoužitelné, se jedná o dobrý pokrok. Dalším výstupem z tohoto experimentu je poznatek, že prohížeč Microsoft Edge výrazně zaostává v rychlosti provádění javascriptu při práci s DOM oproti ostatním prohlížečům.

# 5 Závěr

Hlavním účelem a smyslem této bakalářské práce byl návrh a implementace univerzálně použitelného node editoru. Cílem bylo především usnadnění práce vývojářům nejrůznějších webových aplikací, kteří hodlají ve svém návrhu využít principy node editoru. Cílovou skupinou však vzhledem k jednoduchosti použití navrženého editoru nemusejí být pouze vývojáři či lidé s pokročilou znalostí programování. Lze si představit použití editoru studenty např. pro simulaci elektrických či logických obvodů. Existující programy jsou většinou placené. Stačí snadno nakofigurovat jednotlivé druhy součástek používaných v obvodu a ty pak libovolně pospojovat. Stačí pouze znalost fyzikálních vzorců pro práci s elektřinou a základního programování v javascriptu pro napsání jednoduchých výrazů reflektujících fyzikální pravidla. Jiným příkladem použití může být např. matematická analýza složitějších matematických výpočtů či vizualizace nejrůznějších algoritmů nebo programovacích technik. Zobrazením pomocí node editoru dojde ke zpřehlednění a snadnějšímu pochopení i relativně velmi složitých výpočtů. Možnosti využití tohoto nástroje jsou velmi široké a vzhledem k jejímu dynamickému charakteru jsou omezeny pouze fantazií uživatele.

Samozřejmě existuje spousta námětů na další vylepšení node editoru. Do budoucna by bylo možné promyslet další dynamické vlastnosti aplikace. Např. by se hodila možnost nastavování vzhledu celé aplikace bez nutnosti upravovat css styly samotné aplikace. Pokud bychom chtěli ještě hlubší dynamiku v jádru programu, stálo by za to vymyslet např. způsob přidávání a editace samotných datových typů a práce s nimi tak, aby nebylo nutné zasahovat do jádra editoru.

V poslední části aplikace byly provedeny výkonnostní testy prezentační vrstvy v nejpoužívanějších prohlížečích na jejichž základě potom byly provedeny optimalizace. Tato vylepšení nakonec vedla k poměrně slušnému zlepšení výkonu aplikace i v porovnání s jinými node editory známých programů. Editor se chová poměrně přijatelně i při větším počtu zobrazených uzlů a spojení a složitější aplikační logice. Nejedná se jenom o test samotného node editoru, ale také o test současných prohlížečů, jejich javascriptových interpretů a míry podpory nejnovějších technologií a současných standardů HTML5 a CSS3. Z tohoto srovnání vychází nejlépe prohlížeč Google Chrome těsně následovaný Firefoxem. Naopak zcela propadl prohlížeč Edge, který zatím zaostává jak v rychlosti javascriptového interpreta a práci s DOM, tak v podpoře HTML5 standardů.

# Literatura

[1] CROCKFORD, Douglas. The application/json media type for javascript object notation (json). 2006.

[2] CROCKFORD, Douglas. JavaScript: The Good Parts: The Good Parts. "O'Reilly Media, Inc.", 2008.

[3] PILGRIM, Mark. HTML5: up and running. "O'Reilly Media, Inc.", 2010.

[4] MCFARLAND, David Sawyer. CSS3: the missing manual. "O'Reilly Media, Inc.", 2012.

[5] FULTON, Steve; FULTON, Jeff. HTML5 canvas. "O'Reilly Media, Inc.", 2013.

[6] FAROUKI, Rida T.; RAJAN, V. T. Algorithms for polynomials in Bernstein form. Computer Aided Geometric Design, 1988, 5.1: 1-26.

# A Přílohy

- Obsah CD:
  \Text - Textová část práce, soubor v PDF
  \Program - složka obsahující samotnou aplikaci
    \Data - obsahuje soubor s předdefinovanými nastaveními editoru
    \Scripts - javascriptové soubory s kódem aplikace
    \Styles - soubory s kaskádovými styly a zdrojový less
    \Images - složka s obrázky
    index.html - vstupní bod aplikace, ukázka použití